***CSC 3020***

***Java Programming***

**Assignment 02**

**50 points**

**Due 02/18/2020 (11:45 A.M.)**

Assignment Objectives:

■■ To write programs for executing statements repeatedly using a **while** loop.

■■ To follow the loop design strategy to develop loops.

■■ To control a loop with the user confirmation or a sentinel value.

■■ To write loops using **do-while** statements.

■■ To write loops using **for** statements.

■■ To discover the similarities and differences of three types of loop statements.

■■ To write nested loops.

■■ To implement program control with **break** and **continue.**

■■ To process characters in a string using a loop.

■■ To define methods with formal parameters.

■■ To invoke methods with actual parameters (i.e., arguments).

■■ To define methods with a return value.

■■ To use method overloading and understand ambiguous overloading.

■■ To declare array reference variables and create arrays.

■■ To obtain array size using **arrayRefVar.length** and know default values in an array.

■■ To declare, create, and initialize an array using an array initializer.

■■ To copy contents from one array to another.

■■ To develop and invoke methods with array arguments and return values.

■■ To define a method with a variable-length argument list.

■■ To search elements using the linear or binary search algorithm.

■■ To sort an array using the selection sort approach.

■■ To use the methods in the **java.util.Arrays** class .

■■ To pass arguments to the main method from the command line.

■■ To declare variables for two-dimensional arrays, create arrays, and access array elements in a two-dimensional

array using row and column indices.

■■ To program common operations for two-dimensional arrays (displaying arrays, summing all elements, finding

the minimum and maximum elements, and random shuffling).

■■ To pass two-dimensional arrays to methods.

**Solution to this assignment will not be posted on Canvas; however, any question can be discussed in the class upon request of a student.**

Answer questions 1 to 20 on a word file; write a program for each of Q.21 - Q.23.

All assignments must be submitted by the Canvas. **No email or hard copy** is accepted. You must follow the following format:

1. For non-programming questions, use a word file to type your answers. Don’t use the text box on the Canvas to answer the questions or to write comments, we will not read it.
2. State your answer clearly.
3. For programming questions, include only the source file for each problem.
4. Submit your file to the Canvas. You must submit your assignment on time; otherwise, you will receive zero. In addition, you cannot submit your file more than one time.
5. There will be several folders on the Canvas. You need to upload your file(s) using the correct folder on the Canvas.
6. Name each file: “Assignment Number(Question number(s))”.
7. To upload your file(s):

* In Course Navigation, click the Assignments link.
* Click the title of the assignment.
* Click the **Submit** Assignment button.
* Add **File**. ...
* Add Another **File**. ...
* **Submit** Assignment. ...
* View **Submission**.

**It is your responsibility to make sure that each file is uploaded correctly. If you uploaded a wrong file, you receive zero; files will not be accepted after due date even if you have a prove that the file is created before the due date.**

**Make sure you review the Cheating & Plagiarism policy on Canvas.**

**Q01. (2 points)**

Convert the following for loop statement to a while loop and to a do-while loop:

**long** sum = 0;

**for** (**int** i = 0; i <= 1000; i++)

sum = sum + i;

Solution (while loop):

long sum = 0;

int i = 0;

while(i <= 1000)

sum = sum + i;

Solution (do-while loop):

long sum = 0;

int i = 0;

do {

sum = sum + i;

i++;

} while(i <= 1000);

**Q02. (2 points)**

The for loop in (a) is converted into the while loop in (b). What is wrong? Correct it.

(a)

**int** sum = 0;

**for** (**int** i = 0; i < 4; i++) {

**if** (i % 3 == 0) **continue**;

sum += i;

}

(b)

**int** i = 0, sum = 0;

**while** (i < 4) {

**if** (i % 3 == 0) **continue**;

sum += i;

i++;

}

Solution:

The continue statement creates an infinite loop because it jumps to the next iteration without incrementing i first.

Corrected:

int i = 0, sum = 0;

while (i < 4) {

if (i % 3 == 0) {

i++;

continue;

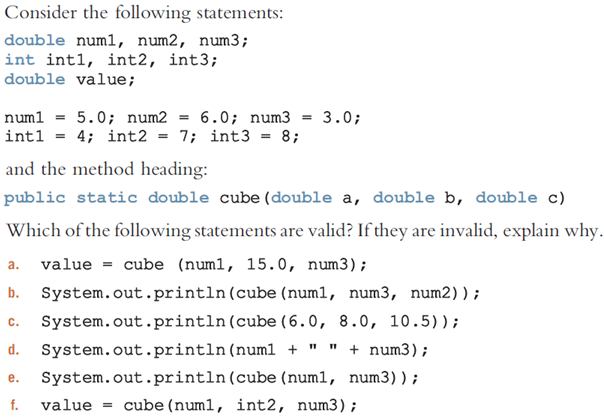
}

sum += i;

i++;

}

**Q 03. (3 points)**



1. **valid.**
2. **valid.**
3. **valid.**
4. **valid.**
5. **invalid, the method cube() takes three arguments.**
6. **invalid, the method cube() takes only doubles**

**Q 04. (0.5 point)**

What is wrong in the following program?

public class Test {

public static void method(int x) {

}

public static int method(int y) {

return y;

}

}

void method was already defined in the first function but never used.

**Q 05. (1.5 points)**

Given two method definitions,

public static double m(double x, double y)

public static double m(int x, double y)

tell which of the two methods is invoked for:

a. double z = m(4, 5);

b. double z = m(4, 5.4);

c. double z = m(4.5, 5.4);

**a. can be either.**

**b. can be either.**

**c. the first one.**

**Q06. (2 points)**

Identify and correct the errors in the following program (4 errors):

1 **public** **class** Test {

2 **public** **static** method1(**int** n, m) {

3 n += m;

4 method2(3.4);

5 }

6

7 **public** **static** **int** method2(**int** n) {

8 **if** (n > 0) **return** 1;

9 **else** **if** (n == 0) **return** 0;

10 **else** **if** (n < 0) **return** -1;

11 }

12 }

1. **public** **class** Test {
2. **// need to change method to int and argument to to int m**

2 **public** **static** method1(**int** n, int m) {

3 n += m;

4 method2(3.4);

5 }

6

7 **public** **static** **int** method2(**double** n) { //need to change to double

8 **if** (n > 0) **return** 1;

9 **else** **if** (n == 0) **return** 0;

10 **else** **if** (n < 0) **return** -1;

11 }

12 }

**Q07. (2.5 points)**

Identify and fix the errors in the following code (5 errors):

1 **public** **class** Test {

2 **public** **static** **void** main(String[] args) {

3 **double**[100] r;

4

5 **for** (**int** i = 0; i < r.length(); i++);

6 r(i) = Math.random \* 100;

7 }

8 }

1 **public** **class** Test {

1. **public** **static** **void** main(String[] args) {
2. **// change to**

5 **double**[] r = new double[100];

6 // remove () after r.length

7 **for** (**int** i = 0; i < r.length; i++);

8 // change r(i) to r[i] add () after .random()

8 r[i] = Math.random() \* 100;

9 }

10 }

**Q08. (0.5 points)**

Once an array is created, its size cannot be changed. Does the following code resize the array?

**int**[] myList;

myList = **new** **int**[10];

// Sometime later you want to assign a new array to myList

myList = **new** **int**[20];

No it overwrites the old array.

**Q09. (0.5 points)**

True or false? When an array is passed to a method, a new array is created and passed to the method.

False

**Q10. (1 points)**

Show the output of the following two programs:

(a)

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** number = 0;

**int**[] numbers = **new** **int**[1];

m(number, numbers);

System.out.println("number is " + number

+ " and numbers[0] is " + numbers[0]);

}

**public** **static** **void** m(**int** x, **int**[] y) {

x = 3;

y[0] = 3;

}

}

(b)

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int**[] list = {1, 2, 3, 4, 5};

reverse(list);

**for** (**int** i = 0; i < list.length; i++)

System.out.print(list[i] + " ");

}

**public** **static** **void** reverse(**int**[] list) {

**int**[] newList = **new** **int**[list.length];

**for** (**int** i = 0; i < list.length; i++)

newList[i] = list[list.length - 1 - i];

list = newList;

}

}

1. **Output: number is 0 and numbers[0] is 3.**
2. **Output: 1 2 3 4 5**

**Q11. (1.5 points)**

What is wrong with each of the following method headers?

**public** **static** **void** print(String... strings, **double**... numbers)

**public** **static** **void** print(**double**... numbers, String name)

**public** **static** **double**... print(**double** d1, **double** d2)

**Q12. (0.5 points)**

What types of array can be sorted using the java.util.Arrays.sort method? Does this sort method create a new array?

One dimensional arrays can be sorted with the java.util.Arrays.sort method. No it does not create a new array.

**Q13. (1 points)**

Show the output of the following code:

**int**[] list1 = {2, 4, 7, 10};

java.util.Arrays.fill(list1, 7);

System.out.println(java.util.Arrays.toString(list1));

**int**[] list2 = {2, 4, 7, 10};

System.out.println(java.util.Arrays.toString(list2));

System.out.print(java.util.Arrays.equals(list1, list2));

Output:

[7, 7, 7, 7]

[2, 4, 7, 10]

false

**Q14. (1.5 points)**

Show the output of the following program when invoked using   
1. java Test I have a dream   
2. java Test "1 2 3"   
3. java Test

**public** **class** Test {

**public** **static** **void** main(String[] args) {

System.out.println("Number of strings is " + args.length);

**for** (**int** i = 0; i < args.length; i++)

System.out.println(args[i]);

}

}

1. **Number of strings is 4**
   1. **I**
   2. **have**
   3. **a**
   4. **dream**
2. **Number of strings is 1**
   1. **1 2 3**
3. **Number of strings is 0**

**Q15. (2 points)**

1. Can the rows in a two-dimensional array have different lengths?
   1. Yes each element of an array is its own 1D array.
2. What is the output of the following code?

**int**[][] array = **new** **int**[5][6];

**int**[] x = {1, 2};

array[0] = x;

System.out.println("array[0][1] is " + array[0][1]);

**Output:** array[0][1] is 2

1. Show the output of the following code:

**int**[][] array = {{1, 2}, {3, 4}, {5, 6}};

**for** (**int** i = array.length - 1; i >= 0; i--) {

**for** (**int** j = array[i].length - 1; j >= 0; j--)

System.out.print(array[i][j] + " ");

System.out.println();

}

**Output:**

**6 5**

**4 3**

**2 1**

1. Show the output of the following code:

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int**[][] array = {{1, 2, 3, 4}, {5, 6, 7, 8}};

System.out.println(m1(array)[0]);

System.out.println(m1(array)[1]);

}

**public** **static** **int**[] m1(**int**[][] m) {

**int**[] result = **new** **int**[2];

result[0] = m.length;

result[1] = m[0].length;

**return** result;

}

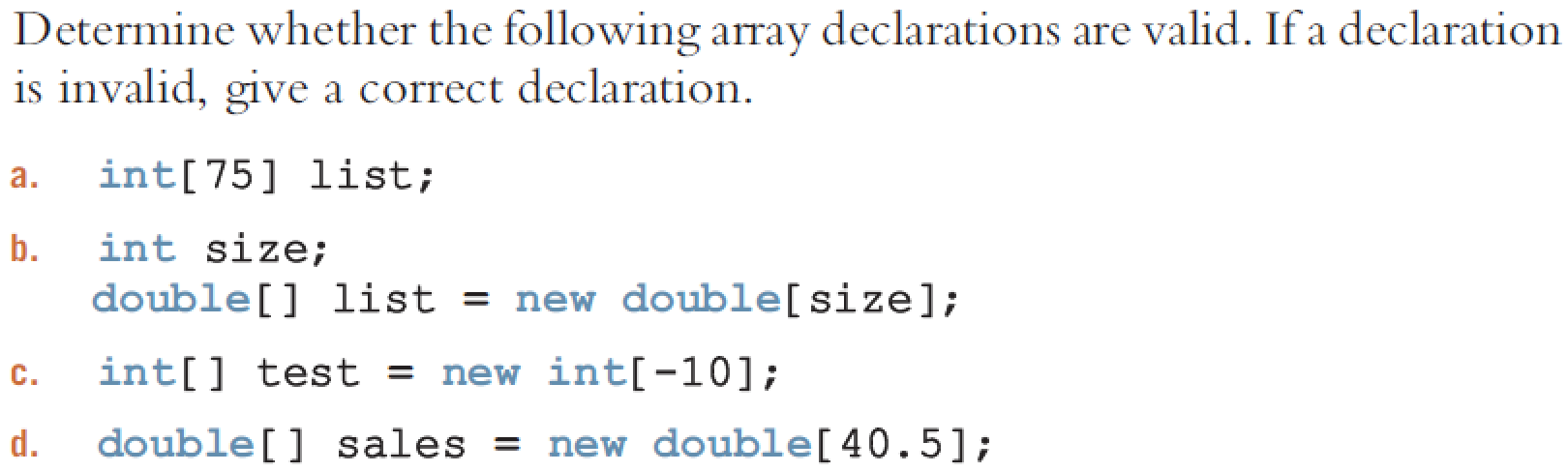
}

**Output:**

**2**

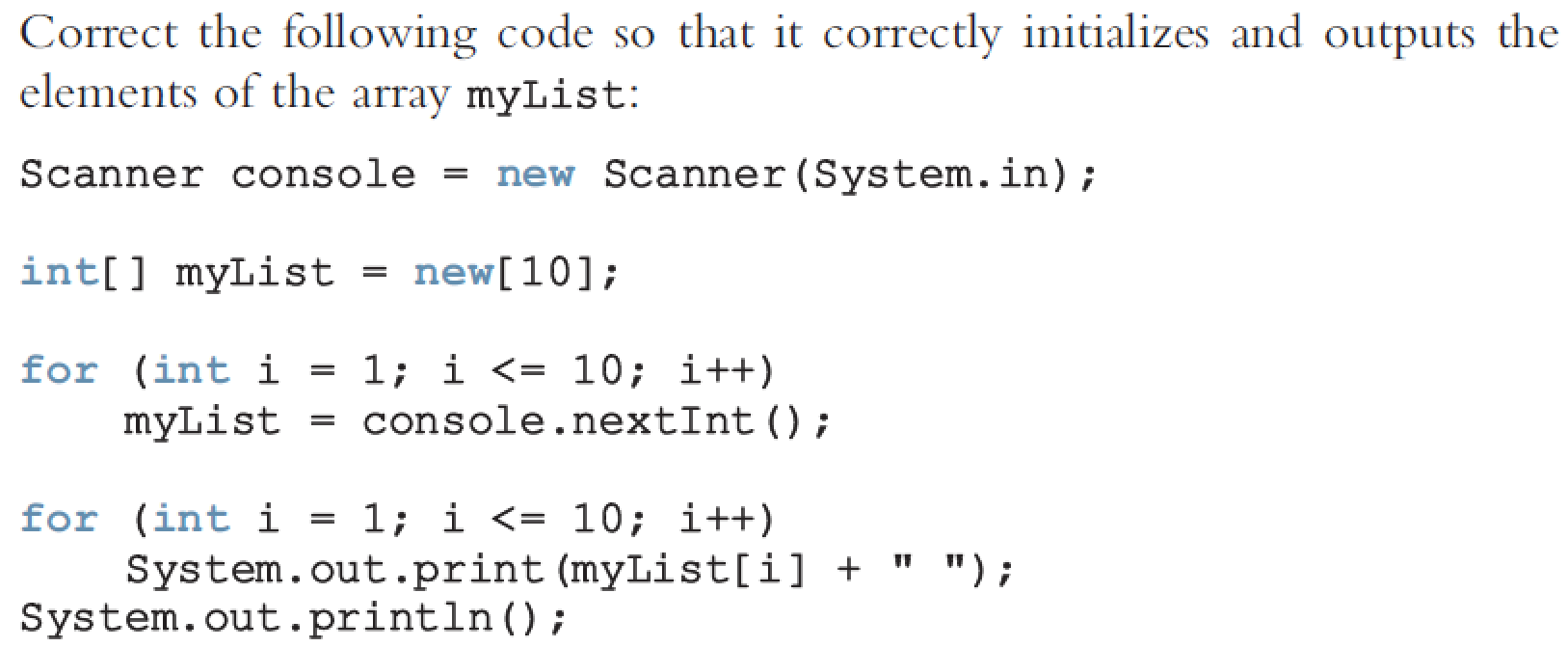
**4**

**Q 16. (2 points)**



* 1. **invalid, int[] list = new int[75];**
  2. **valid;**
  3. **invalid, can’t have a negative array.**
  4. **invalid, can’t have ½ of an array.**

**Q 17. (2 points)**



**Output:**

**Scanner console = new Scanner(System.in);**

**// changed new[10] to new int[10] in order to initialize the array**

**int[] myList = new int[10];**

**// changed to i = 0 and i < 10**

**for (int i = 0; i < 10; i++)**

**// changed from myList to myList[i] so that array could take data**

**myList[i] = console.nextInt();**

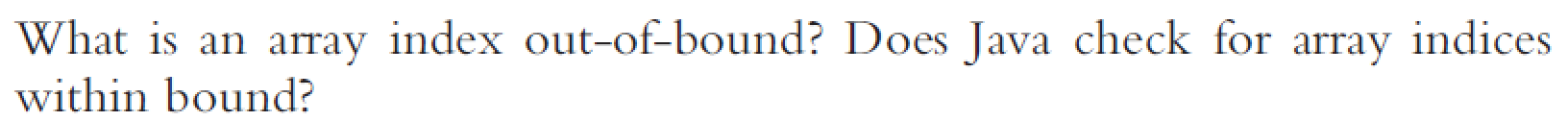
**// changed to i = 0 and i < 10**

**for(int i = 0; i < 10; i++)**

**System.out.print(myList[i] + “ “);**

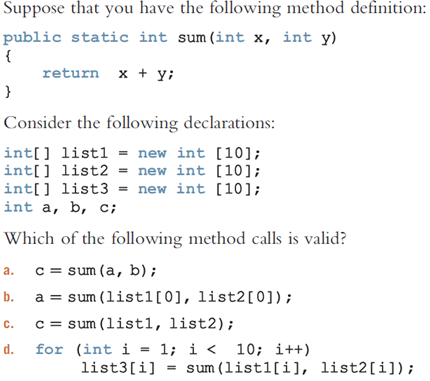
**System.out.println();**

**Q 18. (1 points)**



An array index out-of-bound is an exception that occurs when the code tries to access a nonexisting element outside of the arrays range.

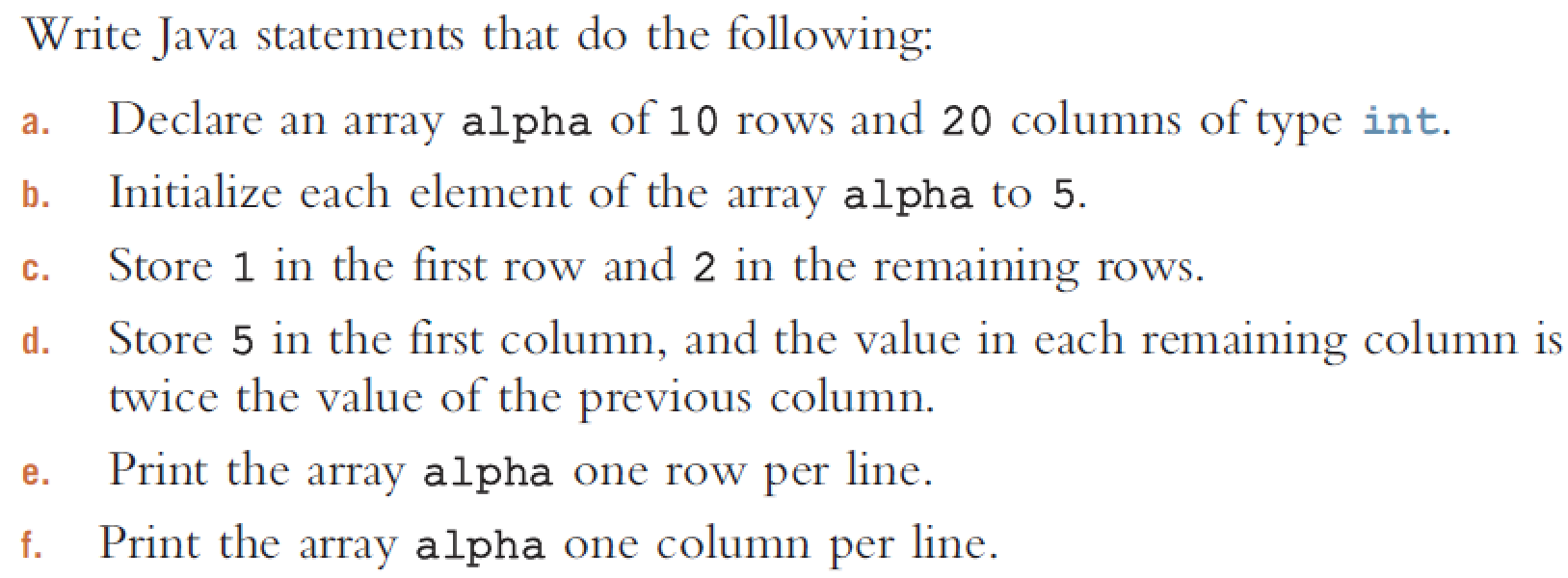
**Q 19. (2 points)**

****

Assume a,b, and c are initialized.

1. **Valid**
2. **Valid**
3. **Invalid, an element must be specified in the array.**
4. **Valid**

**Q 20. (3 points)**



1. **int[][] alpha = new int[10][20];**

**for(int i = 0; i < 10; i++) {**

**for(int j = 0; j < 20; j++) {**

**alpha[i][j] = 5;**

**}**

**}**

**(c)**

**for(int i = 0; i < 10; i++) {**

**for(int j = 0; j < 20; j++) {**

**if(i == 0)**

**alpha[i][j] = 1;**

**else**

**alpha[i][j] = 2;**

**}**

**}**

**(d)**

**for(int i = 0; i < 10; i++) {**

**for(int j = 0; j < 20; j++) {**

**if(j == 0)**

**alpha[i][j] = 5;**

**else**

**alpha[i][j] = alpha[i][j-1] \* 2;**

**}**

**}**

**(e)**

**for(int i = 0; i < 10; i++) {**

**for(int j = 0; j < 20; j++) {**

**if(j == 0)**

**alpha[i][j] = 5;**

**else**

**alpha[i][j] = alpha[i][j-1] \* 2;**

**if(j == 19)**

**System.out.println(alpha[i][j]);**

**else**

**System.out.print(alpha[i][j] + “ “);**

**}**

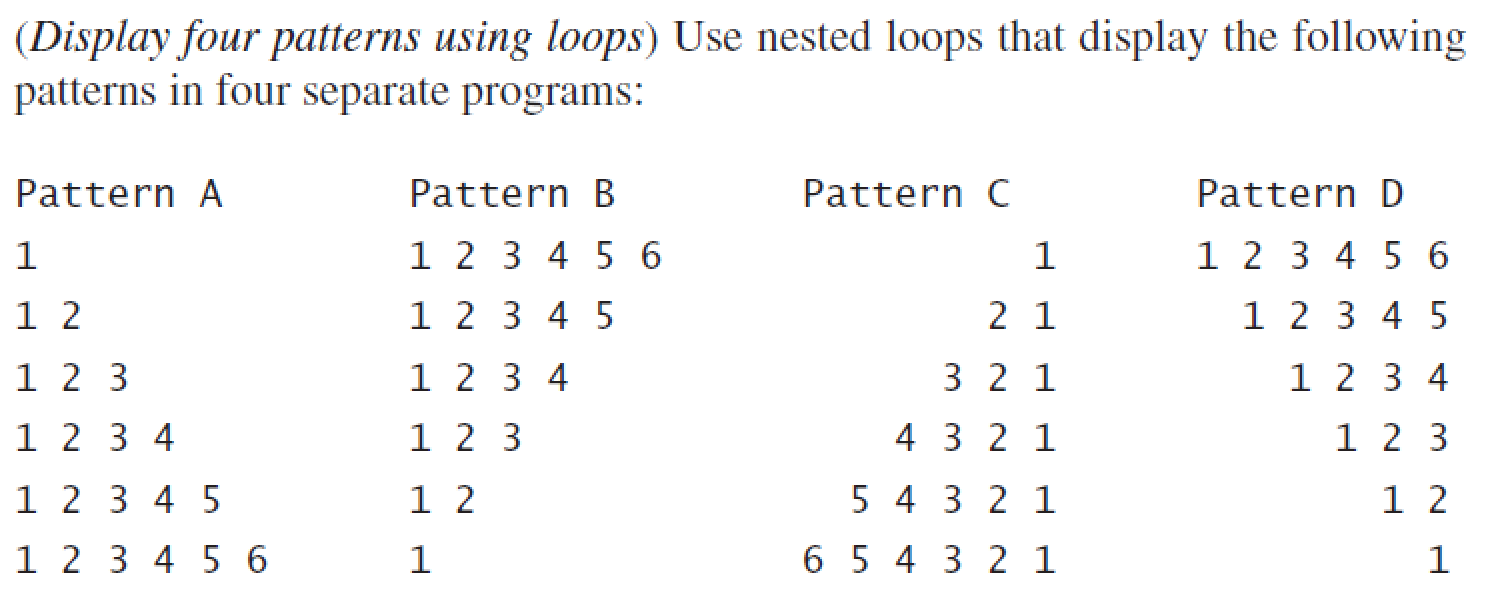
**}**

**(f)**

**for(int i = 0; i < 20; i++) {  
 for(int j = 0; j < 10; j++) {  
 if(j == 9)  
 System.*out*.println(alpha[j][i]);  
 else  
 System.*out*.print(alpha[j][i] + " ");  
 }  
}**

**Programming Questions**

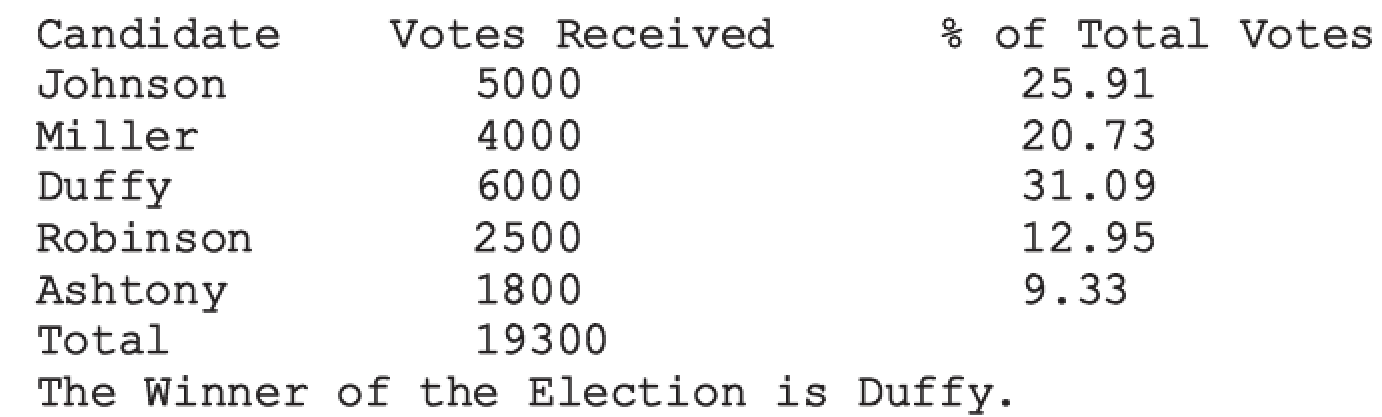
**Q 21. (6 points)**

****

**Q 22. (6 points)**

7. Write a program that allows the user to enter the last names of five candidates in a local election and the votes received by each candidate.

The program should then output each candidate’s name, the votes received by that candidate, and the percentage of the total votes received by the candidate. Your program should also output the winner of the election. A sample output is:



**Q 23. (6 points)**

11. Write a program that uses a two-dimensional array to store the highest and lowest temperatures for each month of the year. The program should output the average high, average low, and highest and lowest temperatures of the year. Your program must consist of the following methods:

1. Method getData: This method reads and stores the data in the two-dimensional array.
2. Method averageHigh: This method calculates and returns the average high temperature of the year.
3. Method averageLow: This method calculates and returns the average low temperature of the year.
4. Method indexHighTemp: This method returns the index of the highest temperature in the array.
5. Method indexLowTemp: This method returns the index of the lowest temperature in the array.

(These methods must all have the appropriate parameters.)

A sample output is:

